Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Севастопольский государственный университет»

ОТЧЕТ

о выполнении лабораторной работы № 3 по дисциплине

«Теория принятия решений»

Выполнил:

ст. гр. ИС/б-22-1-о

Гюнтер М. Ю.

Проверил:

Профессор кафедры

“Информационные системы”

Кротов К. В.

Севастополь,2025

# Цель работы: исследовать применение аппарата теории многомерной полезности при принятии решений по выбору эффективных альтернатив.

# Постановка задачи

Задача состоит в выборе одной из альтернатив, представляющих собой выставленные на продажу автомобили. Критериями (характеристиками) решений являются: и . Используя метод, реализующий построение и исследование двумерной функции полезности, для заданных диапазонов значений критериев и их (критериев) дискретных оценок выполнить: формирование линий безразличия , определение на их основе дискретных значений оценок одномерных функций полезности для каждого из критериев и , аппроксимацию дискретных значений одномерных функций полезности с использованием полиномов второй степени, вычисление коэффициента масштабирования j на основе выбираемых ЛПР по кривым безразличия решениям. С использованием сформированных промежуточных решений выполнить для задаваемых характеристик альтернатив вычисление значений одномерных функций полезности, двумерной функции полезности и реализовать выбор эффективного решения. Выполнить вывод исходных данных, всех промежуточных и конечных результатов. Исходными данными для решаемой задачи являются: параметр "цена" изменяется в диапазоне , параметр "пробег" в диапазоне . Шаг дискретизации первого параметра задан равным 25, шаг дискретизации второго параметра задан равным 20. Соответственно, для первого критерия диапазон изменения его значений задан в виде , для второго критерия диапазон задан в виде . Выбор двух эквивалентных решений на одной из кривых безразличия, сформированных программно, выполнить самостоятельно. Данные, на основании которых выбирается эффективное решение, имеют значения, приведенные в таблице ‎2.1.

Таблица ‎2.1 – Данные для выбора эффективного решения

|  |  |  |
| --- | --- | --- |
| Вариант | Цена | Пробег |
| 1 | 30 | 45 |
| 2 | 50 | 30 |
| 3 | 80 | 20 |
| 4 | 25 | 55 |

При выполнении работы необходимо:

# 2.1. Для введенных диапазонов изменения параметров решений (критериев решений) и соответствующих значений этих критериев реализовать процедуру построения двумерной функции полезности , в которой выполнить определение дискретных значений одномерных функций полезности и для соответствующих критериев (реализовать процедуру формирования значений и .

# 2.2. Выполнить построение линий безразличия для двумерной функции полезности , которые в дальнейшем будут использоваться для определения эквивалентных решений, лежащих на одной из этих линий. Координаты этих решений будут использованы при вычислении коэффициента масштабирования .

# 2.3. Реализовать процедуру аппроксимации полученных дискретных значений одномерных функций полезности и с использованием полиномов второй степени ; , результатом реализации этой процедуры являются коэффициенты этих аналитических кривых , , , .

# 2.4. Выполнить формирование процедуры вычисления значения коэффициента масштабирования j, при реализации которой используются координаты и соответствующих эквивалентных решений и , лежащих на одной кривой безразличия (т.е. в качестве исходных данных для этой процедуры использованы координаты , решений и , выбранных на одной кривой безразличия, сформированной в пункте 2.2.

# 2.5. Для задаваемых в варианте характеристик решений с использованием определенных ранее (процедурой в пункте 2.3) аналитических функций ; реализовать процедуру вычисления значений одномерных функций полезности и , а затем двумерной функции полезности с учетом коэффициента масштабирования . В разрабатываемой процедуре выполнить определение эффективного решения с максимальным значением двумерной функции полезности (передаваемыми в реализуемую процедуру наряду с исходными данными являются параметры , , , .

# 2.6. Выполнить вывод: а) линий безразличия, б) полученных значений одномерных функций полезности и , в) видов аппроксимирующих функций ; , г) значений одномерных и двумерной функций полезности для решений, указанных в варианте задания, д) эффективных решений с максимальным значением двумерной функции полезности .

# Ход работы

## Построение кривых безразличия аналитическим путем

По заданному алгоритму построим кривые безразличия:

1. Пусть и – наименьшие значения оценок соответствующих критериев и . Тогда для решения с координатами (, ) .
2. Дискретно заданы значения критериев : 0.01, 0.013, 0.02, 0.04 и : 0.0125, 0.017, 0.025, 0.05. Отсюда начальное значение критерия . Зададим . Отсюда для решения , соответствующего точке критериального пространства , вычисляем значение многомерной функции полезности . Таким образом получим первую кривую безразличия .
3. Определим значение критерия такое, что , то есть решение , соответствующее точке критериального пространства , эквивалентно известному решению , то есть точки лежат на одной кривой безразличия . Таким значением выберем начальное значение критерия . Тогда значение многомерной функции полезности , отсюда .
4. Для точки критериального пространства , соответствующей решению , определены значения одномерной функции полезности по каждому из критериев: Отсюда значение аддитивной многомерной функции полезности . Получаем, что точка критериального пространства лежит на кривой безразличия .
5. Определим значения критериев и такие, что , то есть решения и , соответствующие точкам критериального пространства и , эквиваленты известному решению , то есть точки лежат на одной кривой безразличия . Такими значениями выберем следующие дискретные значения критериев и 0.013 и 0.017. Тогда , то есть .
6. Для точек критериального пространства и , соответствующих решениям и , определены значения одномерной функции полезности по каждому из критериев: . Также выполняется условие соответсвенных замещений, отсюда , и . Получаем, что точки критериального пространства и лежат на кривой безразличия .
7. Определим значения критериев и такие, что , то есть решения и , соответствующие точкам критериального пространства и , эквиваленты известным решениям и , то есть точки лежат на одной кривой безразличия . Такими значениями выберем следующие дискретные значения критериев и 0.02 и 0.025. Тогда , то есть .
8. Для точек критериального пространства , и , соответствующих решениям , и , определены значения одномерной функции полезности по каждому из критериев: . Также выполняется условие соответсвенных замещений, отсюда , и . Получаем, что точки критериального пространства , и лежат на кривой безразличия .
9. Определим значения критериев и такие, что , то есть решения и , соответствующие точкам критериального пространства и , эквиваленты известным решениям , и , то есть точки лежат на одной кривой безразличия . Такими значениями выберем следующие дискретные значения критериев и 0.04 и 0.05. Тогда , то есть .
10. ,

Итого получаем: , , , , , , . Отсюда максимальным решением является решение с наибольшим значением функции полезности .

## Формирование программного кода процедуры определения значений функции полезности решений

Была написана программа на языке Java, позволяющая определить значения функций полезности решений (листинг ‎3.1).

Листинг ‎3.1 – Программа, позволяющая определить значения функций полезности решений

import static java.lang.System.out;

import java.util.Arrays;

import java.util.Collections;

import java.util.ArrayList;

public class Lab2 {

public static int chooseXnPlus1(int[][] a, ArrayList<Integer> Xn) {

int card = a[0].length;

ArrayList<Integer> options = new ArrayList<>();

for (int x : Xn) {

for (int i = 0; i < card; i++) {

if (!Xn.contains(i) && !options.contains(i) && a[i][x] == 1) {

options.add(i);

}

}

for (int j = 0; j < card; j++) {

if (!Xn.contains(j) && !options.contains(j) && a[x][j] == 1) {

options.add(j);

}

}

}

if (!options.isEmpty()) return Collections.min(options);

return -1;

}

public static ArrayList<Integer> getXnPlus(int[][] a, ArrayList<Integer> Xn, int j) {

int card = a[0].length;

ArrayList<Integer> XnPlus = new ArrayList<>();

for (int i = 0; i < card; i++) {

if (a[i][j] == 1 && Xn.contains(i)) {

XnPlus.add(i);

}

}

return XnPlus;

}

public static ArrayList<Integer> getXnMinus(int[][] a, ArrayList<Integer> Xn, int i) {

int card = a[0].length;

ArrayList<Integer> XnMinus = new ArrayList<>();

for (int j = 0; j < card; j++) {

if (a[i][j] == 1 && Xn.contains(j)) {

XnMinus.add(j);

}

}

return XnMinus;

}

public static double computeU(ArrayList<Integer> XnPlus, ArrayList<Integer> XnMinus, double[] Us) {

if (XnPlus.isEmpty() && !XnMinus.isEmpty()) {

double xDoubleQt = -1;

for (int x : XnMinus) {

if (Us[x] > xDoubleQt)

xDoubleQt = Us[x];

}

double U = xDoubleQt + 1;

return U;

}

if (!XnPlus.isEmpty() && XnMinus.isEmpty()) {

double xQt = Us[XnPlus.get(0)];

for (int x : XnPlus) {

if (Us[x] < xQt)

xQt = Us[x];

}

double U = xQt - 1;

return U;

}

if (!XnPlus.isEmpty() && !XnMinus.isEmpty()) {

ArrayList<Integer> inter = new ArrayList<>(XnPlus);

inter.retainAll(XnMinus);

boolean isInter = !inter.isEmpty();

if (isInter) {

return Us[inter.get(0)];

} else {

double xDoubleQt = -1;

for (int x : XnMinus) {

if (Us[x] > xDoubleQt)

xDoubleQt = Us[x];

}

double xQt = Us[XnPlus.get(0)];

for (int x : XnPlus) {

if (Us[x] < xQt)

xQt = Us[x];

}

double U = (xQt + xDoubleQt) / 2;

return U;

}

}

return -1;

}

public static double[] findUs(int[][] a) {

int card = a[0].length;

double[] Us = new double[card];

ArrayList<Integer> Xn = new ArrayList<>();

ArrayList<Integer> XnPlus;

ArrayList<Integer> XnMinus;

// U(x1) = 0

Us[0] = 0;

// Xn = {x1}

Xn.add(0);

for (int xS = 1; xS < card; xS++) {

int xnPlus1 = chooseXnPlus1(a, Xn);

if (xnPlus1 == -1) {

out.println("Алгоритм не реализуется для заданной матрицы отношения.");

break;

}

XnPlus = getXnPlus(a, Xn, xnPlus1);

XnMinus = getXnMinus(a, Xn, xnPlus1);

Us[xnPlus1] = computeU(XnPlus, XnMinus, Us);

Xn.add(xnPlus1);

}

return Us;

}

public static void main(String[] args) {

int[][] a = new int[][] {

{ 0, 0, 0, 0, 0, 0, 0 },

{ 1, 0, 1, 0, 0, 0, 0 },

{ 0, 0, 0, 1, 0, 0, 1 },

{ 0, 1, 0, 0, 1, 1, 0 },

{ 0, 1, 0, 0, 0, 0, 0 },

{ 1, 0, 0, 1, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 1, 0 },

};

double[] Us = findUs(a);

out.printf("Матрица отношения R: \n%s\n\n",

Arrays.deepToString(a).replace("], ", "]\n").replace("[[", "[").replace("]]", "]"));

out.printf("Полезность каждого решения U(xi): %s", Arrays.toString(Us));

}

}

Результаты работы программы приведены на рисунке ‎3.1.

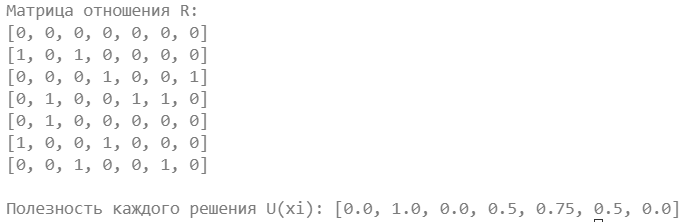


Рисунок ‎3.1 – Результат работы программы

Результаты работы программы говорят о том, что среди решений x1, x2, x3, x4, x5, x6, x7 максимальным решением является , что совпадает с полученным ранее аналитически результатом.

# Вывод:

В результате выполнения лабораторной работы было сформировано базовое представление о функции полезности.